(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
6 October 2011 (06.10.2011)

PCT

(10) International Publication Number
WO 2011/120125 A1

(51) International Patent Classification:
H04L 9/32 (2006.01)

(21) International Application Number:
PCT/CA2010/000486

(22) International Filing Date:
31 March 2010 (31.03.2010)

(25) Filing Language: English

(26) Publication Language: English

(71) Applicant *(for all designated States except US)*: **IRDETO CANADA CORPORATION** [CA/CA]; 84 Hines Road, Suite 300, Ottawa, Ontario K2K 3G3 (CA).

(72) Inventors; and

(75) Inventors/Applicants *(for US only)*: **MUIR, James** [CA/CA]; 82 Willow Glen Drive, Ottawa, Ontario K2M 1T7 (CA). **SUI, Jiayuan** [CN/CA]; 360 Torrance Street, Apt 805, Burlington, Ontario L7R 2R9 (CA). **MURDOCK, Daniel, Elie** [CA/CA]; 156B Valley Stream Drive, Ottawa, Ontario K2H 9C6 (CA). **EISEN, Philip, Allan** [CA/CA]; 170 Nora Street, Ottawa, Ontario K1Z 7B3 (CA).

(74) Agents: **SMITH, Dallas, F.** et al.; Gowling Lafleur Henderson LLP., 160 Elgin Street, Suite 2600, Ottawa, Ontario K1P 1C3 (CA).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PE, PG, PH, PL, PT, RO, RS, RU, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE,

[Continued on next page]

(54) Title: SYSTEM AND METHOD FOR PROTECTING CRYPTOGRAPHIC ASSETS FROM A WHITE-BOX ATTACK

(57) Abstract: A digital signature generation (DSG) process which provides resistance against white box attackers is disclosed. This is done by applying specially selected data transformations to the inputs, outputs and internal parameters of the algorithm. In particular, the signatory's private key does not appear in the clear in our protected implementation. Our new white box implementation produces signatures that are compatible with signatures created by conventional implementations; thus our solution facilitates interoperability and can be used as a drop-in replacement for conventional implementations. In particular, we describe transformations to the key (d) and the generator domain parameter (usually denoted G or g) of the digital signature generation processes, such that embodiments of the invention can produce signed messages which appear to a verifier as if the key (d) was used, without actually ever using the key (d). This makes it impossible for an adversary to ever observe the key (d), as it is not actually used. Further embodiments include additional protections to make it even harder for an adversary to deduce the key (d) by observing the process which generates the digital signature.

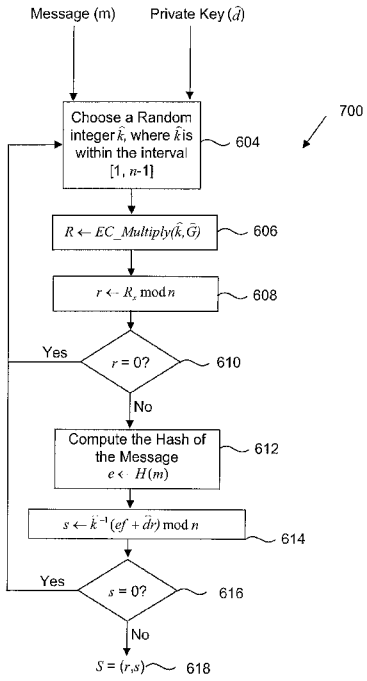

Figure 5
A Robust Process for Generating ECDSA Signatures

WO 2011/120125 A1

ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Declarations under Rule 4.17:

— *as to applicant's entitlement to apply for and be granted a patent (Rule 4.17(ii))*

— *of inventorship (Rule 4.17(iv))*

Published:

— *with international search report (Art. 21(3))*

SYSTEM AND METHOD FOR PROTECTING CRYPTOGRAPHIC ASSETS FROM A WHITE-BOX ATTACK

CROSS REFERENCE TO RELATED APPLICATIONS

[0001] This is the first application for this invention.

FIELD OF THE INVENTION

[0002] The present invention relates generally to cryptography. More particularly, the present invention relates to a method and system for protecting cryptographic assets, such as private keys and other secret parameters.

BACKGROUND OF THE INVENTION

[0003] The Digital Signature Algorithm (DSA) and the Elliptic Curve Digital Signature Algorithm (ECDSA) are described in the standards FIPS PUB 186-3 (U.S. Department of Commerce) and ANS X9.62-2005 (American National Standard for Financial Services), both of which are herein incorporated by reference in their entirety. These signature algorithms use public-key cryptography to enable the creation and verification of digital signatures on digital messages. Signatories in DSA and ECDSA possess a private key and a public key; the private key is used to generate a digital signature (i.e., to sign a message) and the public key is used by third parties to validate that signature.

[0004] DSA and ECDSA are widely deployed (e.g., in ssh, SSL/TLS, Canada Post digital postmarks, DTCP, AACS, MS-DRM) and can be used to provide data origin authentication, data integrity, and non-repudiation. However, any assurances that DSA and ECDSA signatures might provide are always subject to the assumption that a signatory's private key remains private (i.e., the private key does not leak to an attacker).

[0005] The following references provide additional background information, and are each incorporated by reference in their entirety:

[1] American National Standard for Financial Services, ANS X9.62-2005, Public Key Cryptography for the Financial Services Industry, The Elliptic Curve Digital Signature Algorithm (ECDSA), 16 November 2005.

[2] D. Hankerson, A. Menezes, S. Vanstone, Guide to Elliptic Curve Cryptography, 2003.

[3] Information Technology Laboratory, National Institute of Standards and Technology, FIPS PUB 186-3, Digital Signature Standard (DSS), June 2009.

[4] Standards for Efficient Cryptography, SEC 1: Elliptic Curve Cryptography, Version 2.0, 21 May 2009.

[5] National Security Agency, NSA Suite B Cryptography, available from http://www.nsa.gov/ia/programs/suiteb_cryptography/

[6] Digital Transmission Content Protection Specification, Volume 1 (Informational Version), Revision 1.51, 1 October2007.

[0006] The signature generation operation of ECDSA and DSA is typically implemented in computer software, which is then run on a particular computing device (e.g., a cell phone, set-top box, smart card). In many applications, this operation takes place in an environment outside the signatory's control — possibly in the presence of adversaries (i.e., an adversary might observe the device as a signature is being computed).

[0007] An adversary who analyzes only the inputs and outputs of signature generation effectively treats that implementation like a *black box*. DSA and ECDSA were designed to resist such black box attackers. However, there is often more information available than just inputs and outputs. Additional information such as device power consumption, execution time, electromagnetic emanations, and response to data faults can give clues to an attacker about the execution of the software; it has been shown that this can leak bits of the private key and completely compromise the signature scheme.

[0008] A much more robust security model considers resistance against *white box* attackers. White box attackers have full visibility into the execution of the software that computes the signature. Resistance against white box attackers is a highly desired goal, but no white box implementations of DSA or ECDSA have yet been proposed.

As a concrete example of this problem, consider the DTCP protocol used to protect audio/video content. The following quotation comes from the DTCP specification, as defined in reference [6] above:

4.3 Manufacture of Compliant Devices

> All compliant devices that support Full Authentication (that is, each item manufactured, regardless of brand and model) will be assigned a unique Device ID (XID) and device EC-DSA public/private key pair generated by the DTLA. *[The private key] must be stored within the device in such a way as to prevent its disclosure.* Compliant devices must also be given a device certificate (XCERT) by the DTLA. This certificate is stored in the compliant device and used during the authentication process. In addition, the compliant device will need to store the other constants and keys necessary to implement the cryptographic protocols.

[0009] The sentence emphasized above states that DTCP compliant devices must take steps to protect their ECDSA private key. However, such devices must utilize their private key during the Full Authentication protocol to create an ECDSA signature. While it may seem as though the private key can be simply be protected by storing it in an encrypted state, the fact is that, to carry out the signature operation, the private key would first have to be decrypted in order to be used, at which point it could be extracted by a white box attacker. To alleviate this problem what is needed is a protection process that keeps the private key confidential and permits a digital signature operation to be carried out without leaking the private key to the attacker.

SUMMARY OF THE INVENTION

[0010] It is an object of the present invention to obviate or mitigate at least one disadvantage of previous digital signature processes. In particular, one aspect of the invention is directed to a process which more securely protects the key of the signer from being deduced by an adversary who is able to observe the digital signature process in operation.

[0011] Accordingly, aspects of the invention are directed to methods and systems for implementing a digital signature generation (DSG) process which provide resistance against white box attackers. This is done by applying specially selected data transformations to the inputs, outputs and internal parameters of the algorithm. In

particular, the signatory's private key does not appear in the clear in our protected implementation. Our new white box implementation produces signatures that are compatible with signatures created by conventional implementations; thus our solution facilitates interoperability and can be used as a drop-in replacement for conventional implementations. In particular, we describe transformations to the key (d) and the generator domain parameter (usually denoted G or g) of the digital signature generation processes, such that embodiments of the invention can produce signed messages which appear to a verifier as if the key (d) was used, without actually ever using the key (d). This makes it impossible for an adversary to ever observe the key (d), as it is not actually used. Further embodiments include additional protections to make it even harder for an adversary to deduce the key (d) by observing the process which generates the digital signature.

[0012] One aspect of the invention provides a method and system for producing a transformed key ($\hat{d}$), by transforming d based on a random integer f, which can be used along with a transformed generator (e.g., $\hat{G}$ or $\hat{g}$), and f itself, to produce a compatible signed messaged. According to such an aspect, the transformed generator is produced based on the generator, and the inverse of f modulo n. Then the DSG is slightly modified to use said transformed generator (e.g., $\hat{G}$ or $\hat{g}$) and $\hat{d}$, in place of said generator (G or g) and d, along with incorporating f, in said DSG process to produce a digital signature which can be verified by said verification process using the public key Q as if d and said generator were actually used in said DSG process.

[0013] In an aspect, the present invention provides a computer implemented method of protecting a cryptographic secret key (d), which has a corresponding cryptographic public key Q, wherein both Q and d relate to a generator of order n, wherein d is stored and used on a computing apparatus A, from an adversary B able to observe A during execution of a cryptographic digital signature generation (DSG) process which utilizes d, said DSG process having a known signature verification process which utilizes Q, said method comprising:

a) choosing a random integer f in the interval between 1 and n-1;

b) producing a transformed generator based on said generator and the inverse of f modulo n;

c) producing a transformed key ($\hat{d}$) by transforming d based on f; and

d) utilizing said transformed generator and $\hat{d}$, in place of said generator and d, along with incorporating f , in said DSG process to produce a digital signature which can be verified by said verification process using Q as if d and said generator were actually used in said DSG process.

[0014] In a further aspect, there is provided a computer implemented method of protecting a cryptographic secret key (d), which has a corresponding cryptographic public key Q, wherein both Q and d relate to a generator of order n, wherein d is stored and used on a computing apparatus A, from an adversary B able to observe A during execution of a cryptographic digital signature generation (DSG) process which utilizes d, said cryptographic process having a known signature verification process which utilizes Q, said method comprising:

a) choosing a random integer f in the interval between 1 and n-1;

b) producing a transformed generator based on said generator and the inverse of f modulo n;

c) producing a transformed key ($\hat{d}$) by transforming d based on f ; and

d) producing a computer program product embodied in a machine readable medium which stores machine executable instructions, which when executed by A, cause A to utilize said transformed generator and $\hat{d}$, in place of said generator and d, along with adding f , in said DSG process to produce a digital signature which can be verified by said verification process using Q as if d and said generator were actually used in said DSG process.

[0015] In a further aspect, the present invention provides a computer program product comprising a tangible computer readable medium storing computer readable instructions, which when executed by a processor, cause said processor to implement a method of protecting a cryptographic secret parameter (d) which has a corresponding cryptographic public parameter Q, wherein both Q and d relate to a generator of order n, wherein d is stored and used on a computing apparatus A, from an adversary B able to observe A during execution of a cryptographic process which utilizes d, said cryptographic process having a known complimentary process which utilizes Q, said computer readable instructions comprising:

a) instructions for choosing a random integer f in the interval between 1 and n-1;

b) instructions for calculating a transformed generator value of said generator based on an inverse of f mod n

c) instructions for calculating a transformed value $\hat{d}$ of d based on f; and

d) storing said values of f, said transformed generator and $\hat{d}$ in a secure manner for subsequent use in a transformed cryptographic process which uses f, said transformed generator and $\hat{d}$ to produce a digital signature which can be verified by said complimentary process using Q as if d and said generator were actually used in said cryptographic process.

[0016] In further aspect, the present invention provides a computer program product comprising a tangible computer readable medium storing computer readable instructions, which when executed by a computing apparatus (A), cause said computing apparatus to produce a digital signature (r, s) on a message M which can be verified by a standard ECDSA digital signature verification process for verifying a signature (r, s) on a message M purported to have originated with the holder of a private key d, in which said private key d has a corresponding cryptographic public *key* Q, wherein both Q and d relate to a generator G of order n, said signature producible while protecting d from an adversary able to observe A during execution of said computer readable instructions, said computer readable instructions comprising:

i) instructions for choosing a random integer $\hat{k}$ in the interval [1, n − 1].;

ii) instructions for assigning the result of $EC\_Multiply(\hat{k}, \hat{G})$ to a point R, wherein $EC\_Multiply(\hat{k}, \hat{G})$ means add the elliptic curve point $\hat{G}$ to itself $\hat{k}$ times, said elliptic curve point $\hat{G}$ having been previously generated based on a transformation of said generator G using a previously chosen random integer f in the interval between 1 and n-1 ;

iii) instructions for assigning the result of R_x mod n *to* r , where R_x is the x co-ordinate of R;

iv) instructions for branching again to i) if r=0, otherwise;

v) instructions for assigning the value of a hash operation on the message M to e;

vi) instructions for computing the value of $\hat{k}^{-1}(ef + \hat{d}r) \bmod n$ and assign this value to s, wherein $\hat{d}$ is a previously determined transformed value of d based on f;

vii) instructions for branching again to i) if s=0, otherwise;

v) instructions for transmitting the digitally signed message M, along with the signature (r, s) to a 3rd party for verification; and

wherein said signature (r, s) on the message M is equivalent to that which would have been generated by a standard ECDSA process using the private key d, except that the actual private key d is not used and is therefore never loaded into memory during execution. Further, as described herein, according to another aspect, such a computer program product can be adapted to DSA.

[0017] In further aspect, the present invention provides a computer product comprising a tangible computer readable medium storing computer readable instructions, which when executed by a computing apparatus (A), cause said computing apparatus to produce a digital signature (r, s) on a message M using offline transformed values to do the signing, as disclosed herein.

[0018] Other aspects and features of the present invention will become apparent to those ordinarily skilled in the art upon review of the following description of specific embodiments of the invention in conjunction with the accompanying figures.

BRIEF DESCRIPTION OF THE DRAWINGS

[0019] Embodiments of the present invention will now be described, by way of example only, with reference to the attached Figures, wherein:

Fig. 1 is a flowchart illustrating the generic process of digital signature generation.

Fig. 2 is a flowchart illustrating the generic process of digital signature verification.

Fig. 3 is a flowchart illustrating the steps of ECDSA signature generation.

Fig. 4 is a flowchart illustrating the steps of an preliminary procedure, according to an embodiment of the invention.

Fig. 5 is a flowchart illustrating the steps of a process for generating ECDSA signatures according to an embodiment of the invention.

Fig. 6 is a flowchart illustrating the steps of creating a process for making such ECDSA signature generation process more robust according to an embodiment of the invention.

Fig. 7 is a block diagram illustrating a simplified example of a computer system upon which embodiments of the invention may be performed.

DETAILED DESCRIPTION

[0020] Generally, the present invention provides a method and system for protecting cryptographic assets, which include private keys, secret parameters, and the like, in Digital Signature Generation Processes.

[0021] A generic Digital Signature Generation Process is illustrated in Figure 1, in which a message (m) is signed with a private key (d), using a Signature Generation algorithm 104, to produce a Digital Signature (S).

[0022] A generic Digital Signature Verification Process is illustrated in Figure 2, in which a Digital Signature (S) on a message (m), which is purported to have been signed by the holder of the private key (d), is verified using the holder's public key (Q) using a Signature verification algorithm 206. The advantage of such a process is that anyone can use the public key (Q) to verify the signature, and the process will accept or reject 208 the signature, depending on whether it was in fact signed by the holder of the private key (d).

[0023] One aspect of the invention provides methods and systems which can produce a signature (S), while protecting d which is compatible with existing verification processes. In other words, embodiments of the invention will produce S on a computing apparatus A, which may be in the presence of an adversary B who is actively trying to obtain d by observing the signature generation process, which can be verified by 206, using Q, without requiring the verifier change the process 206.

[0024] We describe exemplary processes for converting an implementation of DSA or ECDSA signature generation into a one that resists white box attackers, according to embodiments of the invention, in the following sections.

[0025] Note that we use the term *smooth* to refer to a conventional, unprotected implementation of signature generation (i.e., an implementation that is vulnerable to

white box attackers), and we use the term *robust* to refer to a protected implementation, according to embodiments of the invention.

Notation

[0026] Below we define some terms and notation related to elliptic curve cryptography (ECC) using similar notation described in reference [2]:

[0027]

E	an elliptic curve equation, usually of the form $y^2 = x^3 + ax + b$.
elliptic curve point	an ordered pair (x, y), which satisfies a given equation, E.
a, b	coefficients of an elliptic curve equation.
$\mathbb{F}_p$	a finite field of cardinality p, where p is a prime.
∞	the point-at-infinity
$E(\mathbb{F}_p)$	the elliptic curve group formed by the set of all elliptic curve points, (x, y), along with the point-at-infinity. Note that x and y are elements of $\mathbb{F}_p$.
G	denotes a particular elliptic curve point designated as the generator, or base point. It is publicly known.
G_x, G_y	the x-coordinate and y-coordinate, respectively, of the point G.
kG	the elliptic curve point that results when G is added to itself k times.
$\langle G \rangle$	the subgroup of $E(\mathbb{F}_p)$ generated by the point G.
n	the cardinality of the group $\langle G \rangle$.
$[1, n-1]$	the set of all integers i such that $1 \leq i$ and $i \leq n - 1$.
d	a private key, which is an integer in the interval $[1, n-1]$.
Q	a public key, which satisfies the relation $Q = dG$.
H	a cryptographic hash function (e.g., SHA256).
(r, s)	an ECDSA signature. r and s are integers in the interval $[1, n - 1]$.

Smooth ECDSA

[0028] Here we summarize the unprotected ECDSA Signature Generation operation, which follows the description in ANSX9.62[1].

[0029] Before using ECDSA, a choice must be made on a particular elliptic curve, a cryptographic hash function (H) and a generator point *G*; this information (sometimes referred to as *domain parameters*) is generally considered to be non-secret.

[0030] The following represents the conventional (i.e., smooth) process for signing a message using ECDSA:

Algorithm 1 Smooth ECDSA Signature Generation

Input: a message M, a private key d.

Output: a signature on M consisting of a pair of integers (r, s) in the interval [1, n − 1].

1: choose a random integer k in the interval [1, n − 1].
2: $R \leftarrow kG$.
3: $r \leftarrow R_x \bmod n$. If $r = 0$, then go to Step 1.
4: $e \leftarrow H(M)$.
5: $s \leftarrow k^{-1}(e+dr) \bmod n$. If $s = 0$, then go to Step 1.
6: **return** (r, s)

[0031] Elliptic curve arithmetic is used only in Step 2 to compute the point R; the operations in Steps 3 and 5 are ordinary arithmetic modulo n. Note that the private key, d, is used only at Step 5.

[0032] This process is illustrated in Figure 3. Step 1 as set out above, is labeled as block 304 in that figure. Similarly, Step 2 as set out above, is labeled as block 306 in that figure. Note that as should be readily apparent to a person skilled in the art, step 2 does not involve integer multiplication but rather elliptic curve multiplication, as G is an elliptic curve point. In Step 2, R is assigned (denoted as $\leftarrow$) the point that results when G is added to itself k times, and the notation $EC\_Multiply(k,G)$ is used herein to denote this type of elliptic curve multiplication. Step 3 as set out above, is labeled as block 308 and decision 310, and involves assigning the result of $R_x \bmod n$ to r, where R_x is the x co-ordinate of R, assuming r is non-zero otherwise the process branches back to step 1. Step 4, as illustrated in Fig 3 at block 312, involves assigning to e the value that results when the Hash function is applied to the message; application of the hash function to the message is denoted *H(M)*. Step 5 as set out above, is labeled as block 314 and decision 316, and involves assigning the result of $k^{-1}(e+dr) \bmod n$ to the value s, resulting in the output 318 of the Signature S=(r, s).

[0033] During the above signature generation process, there are actually two highly sensitive values that must be protected:

1. the private key, d.

2. the per-message secret, k.

[0034] That d must be protected is clear. However, if the value of k is leaked, then an adversary can derive d from k. As the hash function H is known, e can be determined from the message M. Accordingly, if an adversary obtains k, an adversary can obtain the private key from the resulting signature (r, s) on M by solving the following equation modulo n:

$$s = k^{-1}(e + dr)$$

Here we have a single equation with d being the only unknown. Thus, d can be solved for. This is a potential security problem.

[0035] Accordingly, an adversary B who observes a computing apparatus A which executes a conventional DSG process, can derive the private key d simply by watching for a call to a random number generator and noting the selection of the value k. Alternatively, an adversary B might observe the instructions executed on apparatus A to compute kG and thereby derive k.

Robust ECDSA

[0036] We now discuss an exemplary process for creating a robust implementation of ECDSA signature generation, according to an embodiment of the invention.

[0037] In some manner not observable by an adversary B, for example during an off-line setup phase, we perform a preliminary process, according to an embodiment. This preliminary process involves selecting a secret number f in the interval [1, $n-1$]. f will be used to protect the private key, d. According to an embodiment, two transformation processes are invoked (both of which are related to f) to produce a transformed (or obfuscated) key and transformed (or obfuscated) generator having the values:

- $\hat{G} = f^{-1}G$, where f^{-1} is computed modulo n.
- $\hat{d} = fd \bmod n$.

[0038] Both $\hat{G}$ and $\hat{d}$ can be computed off-line, immediately following the selection of f.

[0039] This preliminary process, according to an embodiment, is illustrated in Figure 5. In the flowchart of Figure 4, the inputs are the known generator, in this case G, and the private key d. The process starts by choosing 504 a random integer f, where f is within the interval [1, n-1]. The transformed generator $\hat{G}$ is determined 506, by computing the inverse of f modulo n (denoted f^{-1}), and then performing elliptic curve multiplication of f^{-1} and G. Then, the transformed (or obfuscated) key $\hat{d}$ is assigned the value of $fd \bmod n$ at 508. The outputs are the random integer f, the transformed (or obfuscated) key $\hat{d}$ and transformed (or obfuscated) generator $\hat{G}$. All 3 of these values will be used in an embodiment of a Robust ECDSA Signature Generation process, as discussed below.

[0040] Here are the basic steps of a robust signing process, according to an embodiment of the invention:

Algorithm 2 Robust ECDSA Signature Generation

Input: a message M, a private key d.

Output: a signature on M consisting of a pair of integers (r, s) in the interval $[1, n-1]$.

1: choose a random integer $\hat{k}$ in the interval $[1, n-1]$.
2: $R \leftarrow \hat{k}\hat{G}$
3: $r \leftarrow R_x \bmod n$. If $r = 0$, then go to Step 1.
4: $e \leftarrow H(M)$.
5: $s \leftarrow \hat{k}^{-1}(ef + \hat{d}r) \bmod n$. If $s = 0$, then go to Step 1.
6: **return** (r, s)

[0041] As can be seen above, such an embodiment which incorporates a Robust ECDSA Signature Generation process, utilizes the transformed generator $\hat{G}$ and $\hat{d}$, in place of the generator G and d, along with incorporating f (see Step 5), in said ECDSA Signature Generation process to produce a digital signature $S = (r,s)$ which can be verified by a conventional ECDSA verification process using Q as if d and said generator G were actually used in a ECDSA Signature Generation process. In brief, f is incorporated into the process by replacing e with ef in step 5. By utilizing f in Step 5 in this manner --- and recall that f was used in the transformations made offline to produce

$\widehat{G}$ and $\widehat{d}$ --- the resulting signature is fully compatible with one made by a conventional process.

[0042] A flowchart that illustrates the above process, according to an embodiment of the invention is shown in Figure 5. Step 1 in the process 700, as set out above, is labeled as block 604 in that figure. Similarly, Step 2 as set out above, is labeled as block 606 in that figure. Note that as should be readily apparent to a person skilled in the art, step 2 does not involve integer multiplication but rather elliptic curve multiplication, as $\widehat{G}$ is a point on a curve. Accordingly, R is assigned the point that results when $\widehat{G}$ is added to itself $\widehat{k}$ times. Step 3 as set out above, is labeled as block 608 and decision 610, and involves assigning the result of $R_x \mod n$ to the value r, where R_x is the x co-ordinate of R, assuming r is non-zero otherwise the process branches back to step 1. Step 4, as illustrated in Fig 6 at block 612, involves assigning to e the value that results when the hash function H is applied to the message (denoted by $H(M)$). Step 5 as set out above, is labeled as block 614 and decision 616, and involves assigning the result of $\widehat{k}^{-1}(ef + \widehat{d}r) \mod n$ to the value s, resulting in the output 618 of the Signature $S=(r, s)$.

[0043] This version of signature generation produces signatures that are fully compatible with ordinary ECDSA (i.e., any signature (r,s) created by our robust implementation could have also been created by a smooth implementation).

Analysis

[0044] Embodiments of the invention are intended to address the following two goals:

1. Prevent an attacker from being able to deduce the private key from learning the value of the per-message random number $\widehat{k}$. As mentioned previously, if an attacker learns the per-message secret k, then it is possible using a conventional DSG to calculate the secret key d from the message M and its signature (r, s). As explained below, that is not the case here with $\widehat{k}$.
2. Prevent an attacker from learning d. d is the secret key. Note that without other forms of protection the value of $\widehat{d}$ can potentially be observed during the execution

of a signing algorithm by an adversary in a white-box environment. Accordingly, an embodiment adds another layer of protection to $\hat{d}$ by having an additional data transform applied to it. However, the value $\hat{d}$, by itself, leaks no information about d in the following sense: Suppose x is the value of $\hat{d}$. For every integer d in the interval [1, n − 1] there exists a value of f that causes $\hat{d}$ = fd mod n to equal x (i.e., take f equal to xd^{-1} mod n). Thus, is it not possible for an attacker to rule out any possible value of d using only the value of $\hat{d}$.

[0045] To meet the goals above, it does not seem necessary to use additional resources to attempt to add further protection to the computation of R in Step 2. We rationalize this statement as follows.

[0046] Consider the three values used in Step 2: R, $\hat{k}$, $\hat{G}$. Suppose these values are all known to an adversary. Assume further that the attacker knows the value of G since it often appears in publicly available standards. Since $\hat{G}$ = $f^{-1}G$, the adversary might try to compute f^{-1}, and hence f, from this equation, which would contradict goal 2 above. However, this computation amounts to solving an ECDLP, which is assumed to be infeasible. Thus we are confident that an adversary cannot compute f from G and $\hat{G}$.

[0047] Further, regarding goal 1 as set out above, knowledge of $\hat{k}$ does not reveal the secret key d in the same way that knowledge of k does in a conventional ECDSA. This is because the task of computing d from $\hat{k}$ and the signature (r, s) amounts to solving the following equation modulo n:

$$s = \hat{k}^{-1}(e + dr)f$$

[0048] However, unlike for conventional DSGs, the above equation includes two unknown values: d and f (i.e., we have one equation and two unknowns). Therefore the adversary is not able to extract d from this one equation, even if they learn $\hat{k}$, as both d and f are unknown.

[0049] While the embodiments discussed above modify DSGs in such a manner as to make such a process more robust than conventional DSGs, such an improved system can be made even more robust by making it harder for an attacker to learn f.

Accordingly, embodiments of the invention, perform the preliminary steps of choosing f, and producing the transformed values $\hat{G}$ and $\hat{d}$ in some manner not observable by B. One way of performing these steps in a manner not observable by B is to perform these steps offline. By offline, we mean in a time or place such that either A is not observable by B, or on a different computing apparatus altogether. For example, in the case of A being a smart card or set top box, offline includes prior to shipping said device. In the case of A being a personal computer, these steps can be executed on another computer, and provided to A by some secure media. Alternatively, these steps can be executed in a manner not observable B by utilizing secure hardware.

[0050] Further, in a preferred embodiment, additional obfuscation techniques are utilized to make it harder for an adversary to deduce the values of f and $\hat{d}$ from observing the computations in Step 5 (i.e., $s \leftarrow \hat{k}^{-1}(ef + \hat{d}r) \bmod n$). According to one embodiment, step 5 is calculated by executing the following process. First, we create two multiplication functions (which can take the form of software routines obfuscated using a series of look-up tables):

```
mult_by_f(...),   mult_by_dhat(...)
```

[0051] These functions, which according to one embodiment incorporate word-wise recode-tables constructed during the off-line phase, will take a multi-precision integer x as input (i.e., an array of 32-bit words) and return, respectively, $xf \bmod n$ and $x\hat{d} \bmod n$. These two functions will be utilized as follows to carry out the computations of Step 5, according to one embodiment::

5.1: `e_f` $\leftarrow$ `mult_by_f`(e)

5.2: `dhat_r` $\leftarrow$ `mult_by_dhat`(r)

5.3: `tmp` $\leftarrow$ `e_f` + `dhat_r` $\bmod n$

5.4: $s \leftarrow \hat{k}^{-1} *$ `tmp` $\bmod n$

DSA

[0052] Note that for illustrative purposes, we focused above primarily on ECDSA. However ECDSA and DSA are closely related. We submit that one skilled in the art should be able to easily apply the detailed explanation and the examples we have

presented above with respect to ECDSA, to DSA signature generation. However, we now set out a brief summary of the process, according to an embodiment of the invention.

[0053] Here we summarize the unprotected DSA Signature Generation operation, which follows the description in FIPS PUB 186-3 [3].

[0054] Before using DSA, a choice must be made for certain domain parameters: a prime p, a prime q (which divides p − 1), a generator g of order q modulo p. The domain parameters are generally considered to be non-secret.

[0055] Here is how you sign a message using DSA:

Algorithm 3 Smooth DSA Signature Generation

Input: a message M, a private key d.

Output: a signature on M consisting of a pair of integers (r, s) in the interval [1, q − 1].

1: choose a random integer k in the interval [1, q − 1].
2: $r_p \leftarrow g^k . \mathrm{mod} p$.
3: $r \leftarrow r_p \bmod q$. If $r = 0$, then go to Step 1.
4: $e \leftarrow H(M)$.
5: $s \leftarrow k^{-1}(e+dr) \bmod q$. If $s = 0$, then go to Step 1.
6: **return** (r, s)

[0056] Note that the arithmetic operations above are all modular arithmetic (i.e., arithmetic modulo p or arithmetic modulo q)

[0057] During signature generation, there are actually two highly sensitive values that must be protected:

1. the private key, d.
2. the per-message secret, k.

[0058] That d must be protected is clear. If the value of k is leaked, then an adversary can derive d from k and the resulting signature (r, s) on M by solving the following equation modulo q:

$$s = k^{-1}(e + dr)$$

Robust Signature Generation

[0059] One embodiment for creating a robust implementation of a DSA signature generation is as follows.

[0060] Similar to above, in some manner not observable by an adversary B, for example during an off-line setup phase, we select a secret number f in the interval [1, $q-1$]. f will be used to protect the private key, d. According to an embodiment, two transformation processes are invoked (both of which are related to f) to produce a transformed key and transformed generator having the values: f:

- $\hat{g} = g^{f^{-1}} \bmod p$, where f^{-1} is computed modulo q.
- $\hat{d} = fd \bmod q$.

[0061] Both $\hat{g}$ and $\hat{d}$ can be computed off-line, immediately following the selection of f. Here are the basic steps of a robust DSA signing process, according to an embodiment of the invention:

Algorithm 4 Robust DSA Signature Generation

Input: a message M, a private key d.

Output: a signature on M consisting of a pair of integers (r, s) in the interval [1, q − 1].

1: choose a random integer $\hat{k}$ in the interval [1, q − 1].
2: $r_p \leftarrow \hat{g}^{\hat{k}} \bmod p$.
3: $r \leftarrow r_p \bmod q$. If r = 0, then go to Step 1.
4: $e \leftarrow H(M)$.
5: $s \leftarrow \hat{k}^{-1}(ef + \hat{d}r) \bmod q$. If s = 0, then go to Step 1.
6: **return** (r, s)

[0062] This version of signature generation produces signatures that are fully compatible with ordinary DSA (i.e., any signature (r, s) created by our robust implementation could have also been created by a smooth implementation).

Further Protections

[0063] Regardless of which of the above embodiments is implemented, control flow transformations and data transformations can be applied to the source code that implements the signature generation operation, in order to make it harder for an adversary to deduce sensitive parameters, for example $\hat{d}$ or f, by observing the operation of the computing device which executes the DSG process. Figure 7 is a flowchart which illustrates an exemplary process for further protecting the parameters, by obfuscating the software which generates the signature, according to an embodiment of the invention. Accordingly, the source code which implements the process 700 described above, can be further protected by applying control flow transformations 702 and data transformations 704 to produce a more robust implementation.

[0064] The control flow transformations 702 map the control-flow of a given program into a new control-flow form, using a number of functions to control the flow, and include the exemplary control-flow transformations as described in US Patent No. 6,779,114, issued August 17, 2004, which is incorporated herein by reference in its entirety). The data transformations 704 involve the use of mathematical mapping functions which transform both data operations and locations to alternate mathematical spaces, examples of which are described in US Patent No. 6,594,761, issued July 15, 2003, US Patent No. 6,842,862 issued January 11, 2005, and US Patent Publication No. US-2005-0166191A1, published July 28, 2005, the contents of which are incorporated herein by reference in their entirety.

[0065] In another embodiment, to offer another level of protection for f we compute ef mod n (which is used in Step 5) in another manner. Since the value e *is* equal to the hash of the input message, M, an embodiment of the invention incorporates the multiply by f modulo n operation into the hash computation. As will be appreciated, this depends on the nature of the hash function used, but assuming a suitable hash function, in such an embodiment the hash function H is replaced with a function H_f which produces as a result the value $\hat{e} = H_f(M) \bmod n = e * f \bmod n$. Further, in step 5, the term ef is replaced with $\hat{e}$, such that $\hat{k}^{-1}(ef + \hat{d}r) \bmod n$ is replaced with $\hat{k}^{-1}(\hat{e} + \hat{d}r) \bmod n$.

[0066] It should be understood that the present invention may be practiced upon any given computer system. A simplified example of a computer system upon which an embodiment of the invention may be performed is presented as a block diagram in **FIGURE 7**. This computer system **110** includes a display **112**, keyboard **114**, computer **116** and external devices **118**.

[0067] The computer **116** may contain one or more processors or microprocessors, such as a central processing unit (CPU) **120**. The CPU **120** performs arithmetic calculations and control functions to execute software stored in an internal memory **122**, preferably random access memory (RAM) and/or read only memory (ROM), and possibly additional memory **124**. The additional memory **124** may include, for example, mass memory storage, hard disk drives, floppy disk drives, magnetic tape drives, compact disk drives, program cartridges and cartridge interfaces such as those found in video game devices, removable memory chips such as EPROM or PROM, or similar storage media as known in the art. This additional memory **124** may be physically internal to the computer **116**, or external as in **FIGURE 7**.

[0068] The computer system **110** may also include other similar means for allowing computer programs or other instructions to be loaded. Such means can include, for example, a communications interface **126** which allows software and data to be transferred between the computer system **110** and external systems. Examples of communications interface **126** can include a modem, a network interface such as an Ethernet card, a serial or parallel communications port. Software and data transferred via communications interface **126** are in the form of signals which can be electronic, electromagnetic, and optical or other signals capable of being received by communications interface **126**. Multiple interfaces, of course, can be provided on a single computer system **110**.

[0069] Input and output to and from the computer **116** is administered by the input/output (I/O) interface **128**. This I/O interface **128** administers control of the display **112**, keyboard **114**, external devices **118** and other such components of the computer system **110**.

[0070] The invention is described in these terms for convenience purposes only. It would be clear to one skilled in the art that the invention may be applied to other computing apparatus or control systems **110**. Thus the term computer apparatus is intended to include a variety of systems including all manner of appliances having computer or processor control including telephones, cellular telephones, televisions,

television set top units, point of sale computers, automatic banking machines, lap top computers, servers, personal digital assistants and automobiles. Such computer apparatus may include additional components, or omit some of the components discussed above with reference to figure 7,

[0071] Further it should be appreciate that the "offline" portions could be executed on a different computing apparatus than the computing apparatus which performs the signing operation. Further, the computing apparatus which executes the verification process will usually be different than the computing apparatus which executes the digital signing process, and the signed message is often transmitted via some medium. So an aspect of the invention is directed to a system comprising a signing computer apparatus for generating and transmitting a digital signature, and a verifying computer apparatus for verifying the signature. Further, an embodiment could include an additional computer apparatus for executing the preliminary steps of choosing f and producing the transformed key and transformed generator based on f.

[0072] In the preceding description, for purposes of explanation, numerous details are set forth in order to provide a thorough understanding of the embodiments of the invention. However, it will be apparent to one skilled in the art that these specific details are not required in order to practice the invention. In other instances, well-known electrical structures and circuits are shown in block diagram form in order not to obscure the invention. For example, specific details are not provided as to whether the embodiments of the invention described herein are implemented as a software routine, hardware circuit, firmware, or a combination thereof.

[0073] Embodiments of the invention can be represented as a software product stored in a machine-readable medium (also referred to as a computer-readable medium, a processor-readable medium, or a computer usable medium having a computer-readable program code embodied therein). The machine-readable medium can be any suitable tangible medium, including magnetic, optical, or electrical storage medium including a diskette, compact disk read only memory (CD-ROM), memory device (volatile or non-volatile), or similar storage mechanism. The machine-readable medium can contain various sets of instructions, code sequences, configuration information, or other data, which, when executed, cause a processor to perform steps in a method according to an embodiment of the invention. Those of ordinary skill in the art will appreciate that other instructions and operations necessary to implement the described invention can

also be stored on the machine-readable medium. Software running from the machine-readable medium can interface with circuitry to perform the described tasks.

[0074] The above-described embodiments of the invention are intended to be examples only. Alterations, modifications and variations can be effected to the particular embodiments by those of skill in the art without departing from the scope of the invention, which is defined solely by the claims appended hereto.

CLAIMS:

1. A computer implemented method of protecting a cryptographic secret key (d), which has a corresponding cryptographic public *key* Q, wherein both Q and d relate to a generator of order n, wherein d is stored and used on a computing apparatus A, from an adversary B able to observe A during execution of a cryptographic digital signature generation (DSG) process which utilizes d, said DSG process having a known signature verification process which utilizes Q, said method comprising:

a) choosing a random integer f in the interval between 1 and n-1;

b) producing a transformed generator based on said generator and the inverse of f modulo n;

c) producing a transformed key ($\hat{d}$) by transforming d based on f; and

d) utilizing said transformed generator and $\hat{d}$, in place of said generator and d, along with incorporating f, in said DSG process to produce a digital signature which can be verified by said verification process using Q as if d and said generator were actually used in said DSG process.

2. The method as claimed in claim 1 wherein steps a, b, and c are executed in some manner not observable by B and wherein step d is executed by A in conditions which could be observed by B.

3. The method as claimed in claim 2, wherein steps a, b, and c are executed in some manner not observable by B by executing steps a, b, and c offline.

4. The method as claimed in any one of claims 1-3, wherein said generator is a point G on an elliptic curve used to generate an elliptic curve group having a cardinality of n, and wherein said transformed generator is another point $\hat{G}$ in said elliptic curve group and wherein step b) comprises assigning to f^{-1} the inverse of f modulo n, and assigning to $\hat{G}$ the result of adding G to itself f^{-1} times; and wherein step c) comprises assigning to $\hat{d}$ the result of $d\,f \bmod n$.

5. The method as claimed in claim 4 wherein said digital signature verification process is a standard ECDSA digital signature verification process for verifying a signature (r, s) on a message M originated with the holder of a private key d and wherein step d) comprises:

i) Choose a random integer $\hat{k}$ in the interval [1, n − 1];

ii) Assign the result of $EC\_Multiply(\hat{k}, \hat{G})$ to a point R, wherein $EC\_Multiply(\hat{k}, \hat{G})$ means add the elliptic curve point $\hat{G}$ to itself $\hat{k}$ times;

iii) Assign the result of $R_x \bmod n$ to r, where R_x is the x co-ordinate of R;

iv) if r=0, then start again from step i), otherwise;

v) assign the value of a hash operation on the message M to e;

vi) compute the value of $\hat{k}^{-1}(ef + \hat{d}r) \bmod n$ and assign this value to s;

vii) if s=0, then start again from step i), otherwise;

v) output the signature (r, s)

wherein said signature (r, s) on the message M is equivalent to that which would have been generated by a standard ECDSA process using the private key d, except that the actual private key d is not used and is therefore never loaded into memory during the execution of step d

6. The method as claimed in any one of claims 1-3, wherein digital signature verification process is a standard DSA digital signature verification process for verifying a signature (r,s) on a message M originated with the holder of a private key d and which utilizes domain parameters p, q, and g, wherein said generator is g and wherein q=n and wherein step b) comprises assigning to f^{-1} the inverse of f modulo q, and assigning to $\hat{g}$ the result of $g^{f^{-1}} \bmod p$, where f^{-1} is computed modulo q and wherein step c) comprise assigning to $\hat{d}$ the result of d f mod q.

7. The method as claimed in claim 6 wherein said step d) comprises:

i) Choose a random integer $\hat{k}$ in the interval [1, n − 1];

ii) Assign the result of $\hat{g}^{\hat{k}} \bmod p$ to r_p;

iii) Assign the result of r_p mod q to r;

iv) if r=0, then start again from step i), otherwise;

v) assign the value of a hash operation on the message M to e;

vi) compute the value of $\hat{k}^{-1}(ef + \hat{d}r)$ mod n and assign this value to s;

vii) if s=0, then start again from step i), otherwise;

v) output the signature (r, s)

wherein said signature (r, s) on the message M is equivalent to that which would have been generated by a standard DSA process using the private key d, except that the private key d (is not actually used and is therefore never loaded into memory during the execution of step d.

8. The method as claimed in claim 5 or claim 7 wherein step vi) comprises utilizing a software routine which is obfuscated using a series of look-up tables to calculate the value of ef modulo n such that the value of f is not loaded into memory and hence are not revealed during the calculation.

9. The method as claimed in claim 8 wherein step vi) further comprises utilizing a software routine which is obfuscated using a series of look-up tables to calculate the value $\hat{d}r$ modulo n such that neither of the values of f and $\hat{d}$ are loaded into memory and hence are not revealed during the calculation.

10. The method as claimed in claim 8 or 9 wherein said software routines incorporate word-wise recode-tables constructed during the off-line phase.

11. The method as claimed in any one of claims 5 or 7 in which the Hash function H is replaced with a function H_f which produces as a result the value $\hat{e}$ = H_f (M) = e*f mod n, and in which the term ef is replace with $\hat{e}$, such that $\hat{k}^{-1}(ef + \hat{d}r)$ mod n in step iv) is replaced with $\hat{k}^{-1}(\hat{e} + \hat{d}r)$ mod n.

12. The method as claimed in any one of claims 1-11 further comprising control flow and data obfuscations performed on software which executes the steps of said method.

13. A computer implemented method of protecting a cryptographic secret key (d), which has a corresponding cryptographic public *key* Q, wherein both Q and d relate to a *generator* of order n, wherein d is stored and used on a computing apparatus A, from an adversary B able to observe A during execution of a cryptographic digital signature generation (DSG) process which utilizes d, said cryptographic process having a known *signature verification* process which utilizes Q, said method comprising:

a) choosing a random integer f in the interval between 1 and n-1;

b) producing a transformed generator based on said generator and the inverse of f modulo n;

c) producing a transformed key ($\hat{d}$) by transforming d based on f; and

d) producing a computer program product embodied in a machine readable medium which stores machine executable instructions, which when executed by A, cause A to utilize said transformed generator and $\hat{d}$, in place of said generator and d, along with adding f, in said DSG process to produce a digital signature which can be verified by said verification process using Q as if d and said generator were actually used in said DSG process.

14. The method as claimed in claim 13, wherein said digital signature verification process is a standard ECDSA digital signature verification process for verifying a signature (r, s) on a message M originated with the holder of a private key d; and

wherein said generator is a point G on an elliptic curve used to generate an elliptic curve group having a cardinality of n, and wherein said transformed generator is another point $\hat{G}$ in said elliptic curve group and wherein step b) comprises assigning to f^{-1} the inverse of f modulo n, and assigning to $\hat{G}$ the result of adding G to itself f^{-1} times; and wherein step c) comprises assigning to $\hat{d}$ the result of d f mod n.

15. The method as claimed in claim 13 wherein digital signature verification process is a standard DSA digital signature verification process for verifying a signature (r,s) on a message M originated with the holder of a private key d and which utilizes domain parameters p, q, and g, wherein said generator is g and wherein q=n and wherein step

b) comprises assigning to f^{-1} the inverse of f modulo q, and assigning to $\hat{g}$ the result of $g^{f^{-1}} \bmod p$, where f^{-1} is computed modulo q and wherein step c) comprise assigning to $\hat{d}$ the result of d f mod q.

16. A computer program product comprising a tangible computer readable medium storing computer readable instructions, which when executed by a processor, cause said processor to implement a method of protecting a cryptographic secret parameter (d) which has a corresponding cryptographic public parameter Q, wherein both Q and d relate to a generator of order n, wherein d is stored and used on a computing apparatus A, from an adversary B able to observe A during execution of a cryptographic process which utilizes d, said cryptographic process having a known complimentary process which utilizes Q, said computer readable instructions comprising:

a) instructions for choosing a random integer f in the interval between 1 and n-1;

b) instructions for calculating a transformed generator value of said generator based on an inverse of f mod n

c) instructions for calculating a transformed value $\hat{d}$ of d based on f; and

d) storing said values of f, said transformed generator and $\hat{d}$ in a secure manner for subsequent use in a transformed cryptographic process which uses f, said transformed generator and $\hat{d}$ to produce a digital signature which can be verified by said complimentary process using Q as if d and said generator were actually used in said cryptographic process.

17. A computer program product comprising a tangible computer readable medium storing computer readable instructions, which when executed by a computing apparatus (A), cause said computing apparatus to produce a digital signature (r,s) on a message M which can be verified by a standard ECDSA digital signature verification process for verifying a signature (r,s) on a message M purported to have originated with the holder of a private key d, in which said private key d has a corresponding cryptographic

public key Q, wherein both Q and d relate to a generator G of order n, said signature producible while protecting d from an adversary able to observe A during execution of said computer readable instructions, said computer readable instructions comprising:

i) instructions for choosing a random integer $\hat{k}$ in the interval [1, n − 1];

ii) instructions for assigning the result of $EC\_Multiply(\hat{k}, \hat{G})$ to a point R, wherein $EC\_Multiply(\hat{k}, \hat{G})$ means add the elliptic curve point $\hat{G}$ to itself $\hat{k}$ times, said elliptic curve point $\hat{G}$ having been previously generated based on a transformation of said generator G using a previously chosen random integer f in the interval between 1 and n-1;

iii) instructions for assigning the result of $R_x \mod n$ to r, where R_x is the x co-ordinate of R;

iv) instructions for branching again to i) if r=0, otherwise;

v) instructions for assigning the value of a hash operation on the message M to e;

vi) instructions for computing the value of $\hat{k}^{-1}(ef + \hat{d}r) \mod n$ and assign this value to s, wherein $\hat{d}$ is a previously determined transformed value of d based on f;

vii) instructions for branching again to i) if s=0, otherwise;

v) instructions for transmitting the digitally signed message M, along with the signature (r,s) to a 3rd party for verification; and

wherein said signature (r,s) on the message M is equivalent to that which would have been generated by a standard ECDSA process using the private key d, except that the actual private key d is not used and is therefore never loaded into memory during execution.

18. A computer product comprising a tangible computer readable medium storing computer readable instructions, which when executed by a computing apparatus (A), cause said computing apparatus to produce a digital signature (r, s) on a message M using offline transformed values to do the signing, as disclosed herein.

19. A set top box implementing any of the methods and systems as taught or claimed herein.

20. A computer apparatus implementing any of the methods and systems as taught or claimed herein.

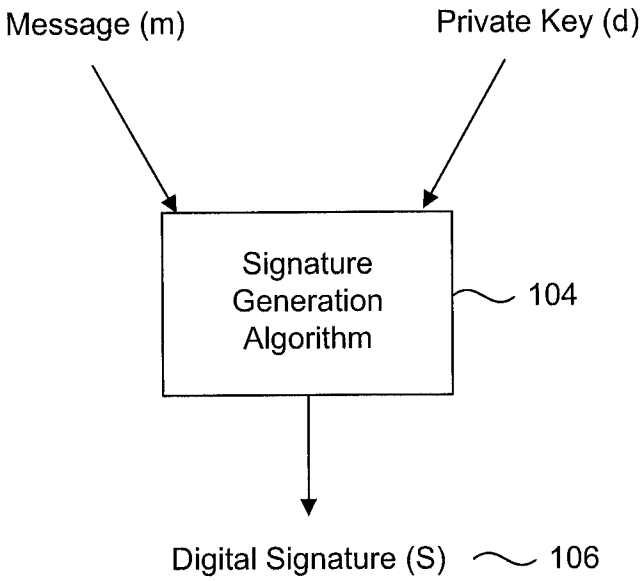

Figure 1

Generation of a Digital Signature

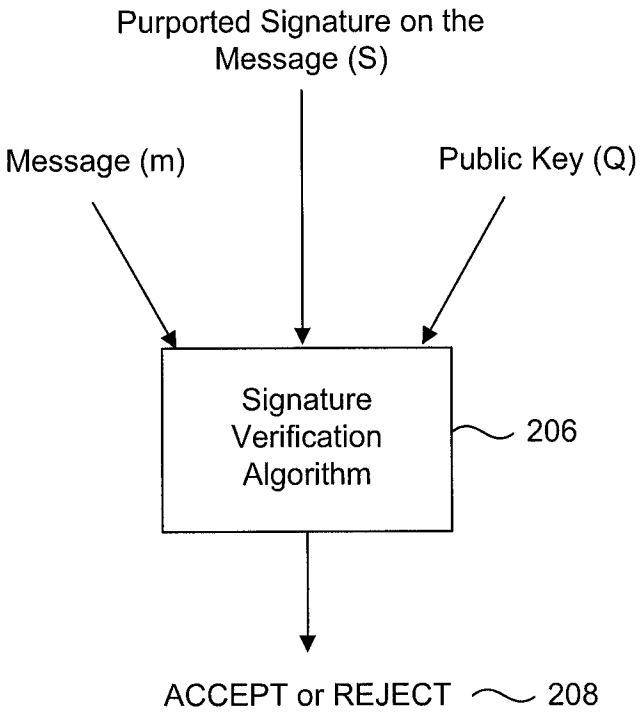

Figure 2

Verification of a Digital Signature

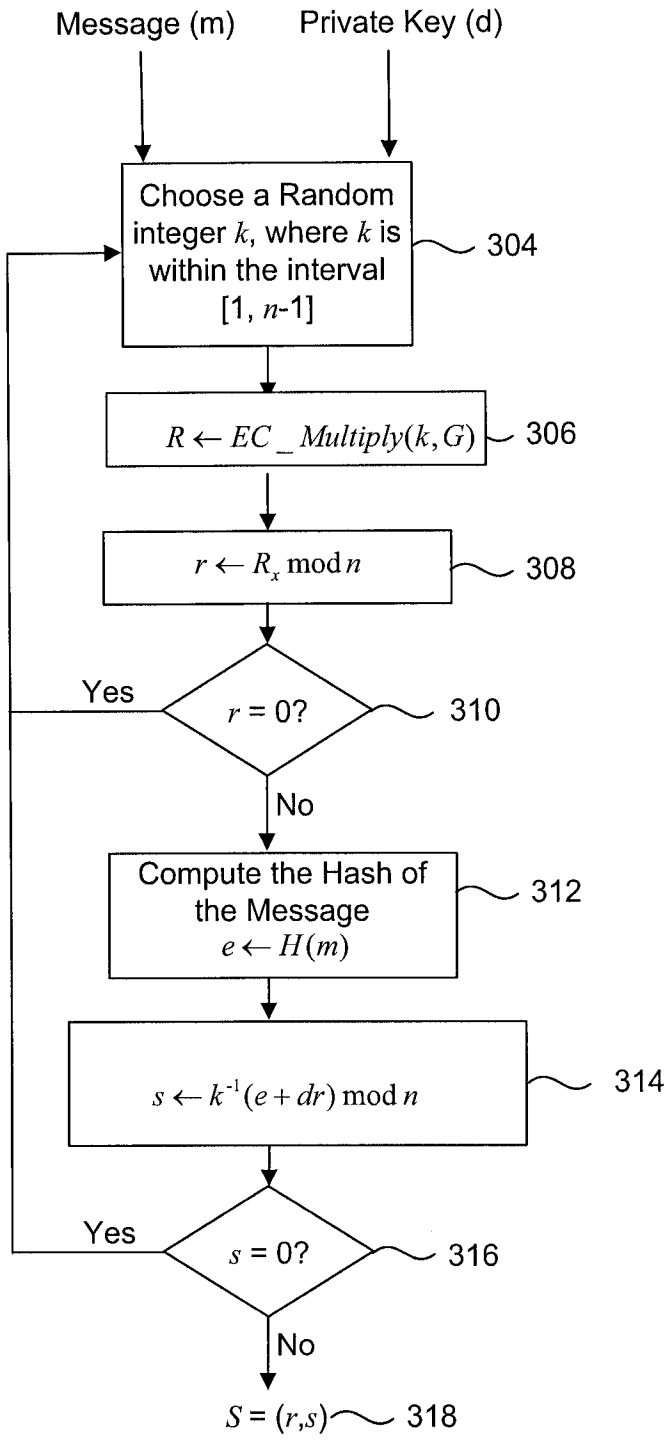

Figure 3

ECDSA Signature Generation

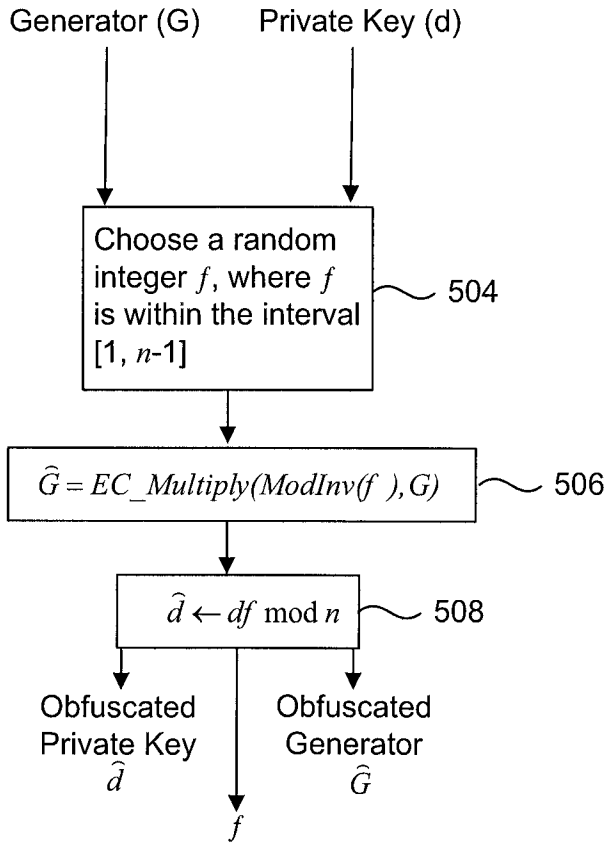

Figure 4

Obfuscation of the Generator (G) and Private Key (d)

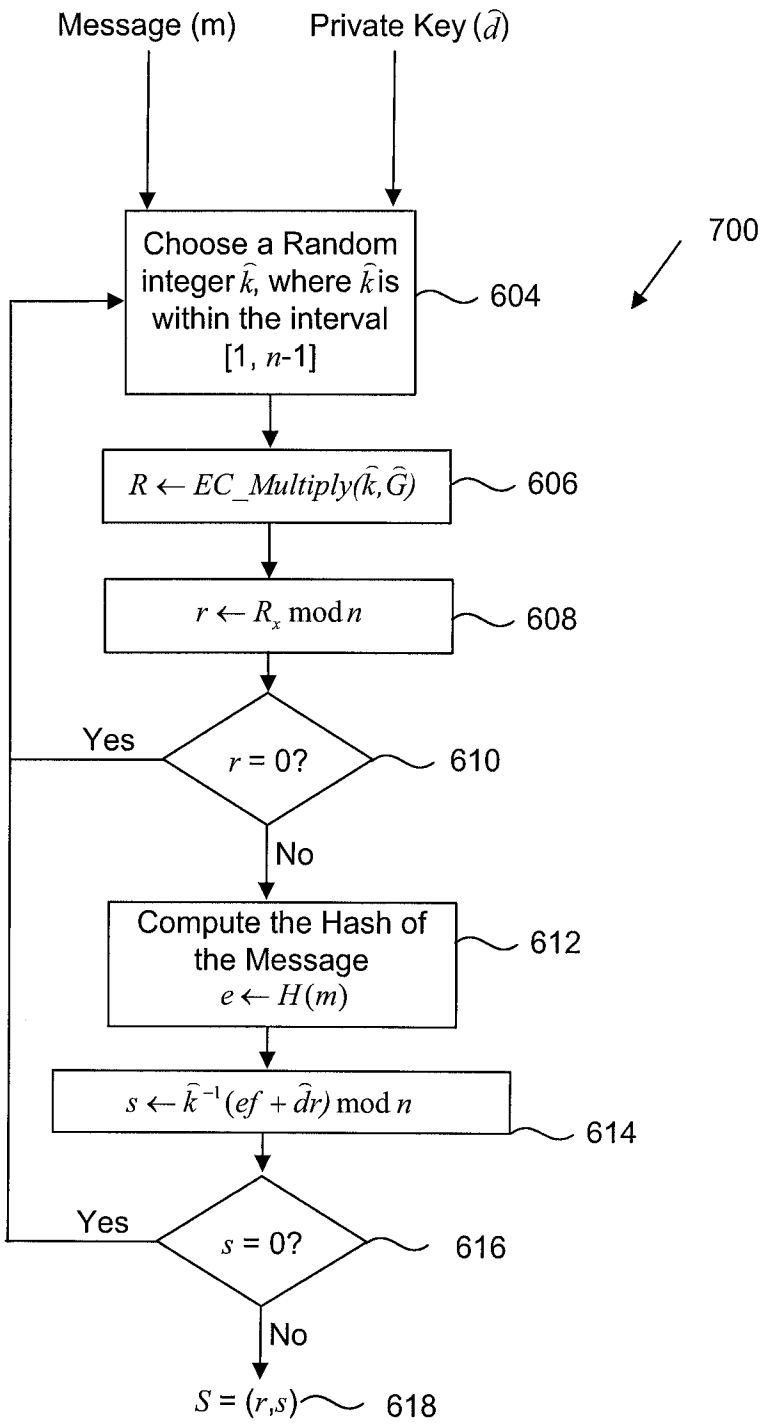

Figure 5
A Robust Process for Generating ECDSA Signatures

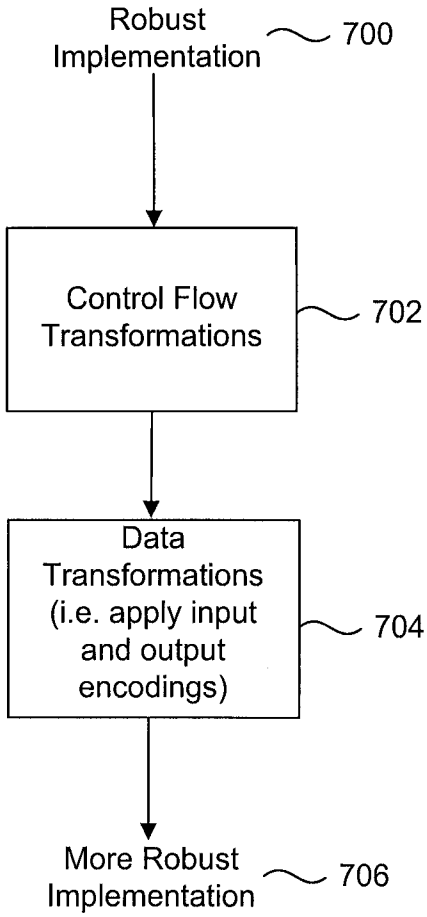

Figure 6

Creation of a More Robust Process for Generating ECDSA Signatures

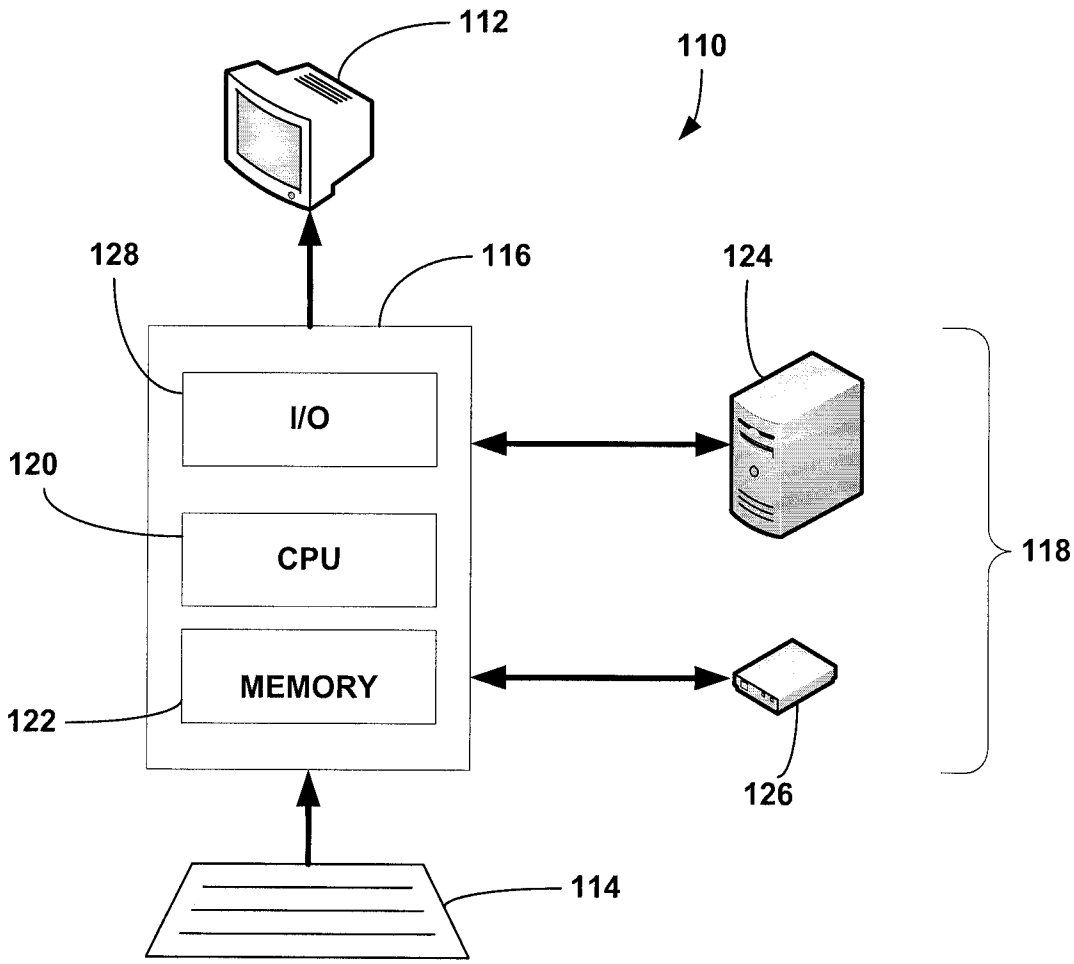

Figure 7

INTERNATIONAL SEARCH REPORT

International application No.
PCT/CA2010/000486

A. CLASSIFICATION OF SUBJECT MATTER
IPC: ***H04L 9/32*** (2006.01)
According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC: ***H04L 9/32*** (2006.01) using keywords

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic database(s) consulted during the international search (name of database(s) and, where practicable, search terms used)
Epoque, Canadian Patent Database, TotalPatent, IEEE Xplore, WEST
Keywords: digital signature algorithm (DSA), elliptic curve digital signature algorithm (ECDSA), white box, cryptanalysis, off-line, private key, obfuscat*, software security, key hiding, software protection, signature generation, piracy prevention

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	US20090252327 A1 "Combination White Box/Black Box Cryptographic Processes and Apparatus"; CIET et al.; 8 October 2009(08-10-2009) ***whole document***	1-20
A	US20060140401 A1 "System and Method for Protecting Computer Software from a White Box Attack"; JOHNSON et al.; 29 June 2006 (29-06-2006) ***whole document***	1-20
A	US20050002532 A1 "System and Method of Hiding Cryptographic Private Keys; ZHOU et al.; 6 Jan 2005 (06-01-2005) ***whole document***	1-20
A	US6668325 B1 "Obfuscation techniques for enhancing software security"; COLLBERG et al.; 23 December 2003 (23-12-2003) ***whole document***	1-20

[ ] Further documents are listed in the continuation of Box C.

[X] See patent family annex.

* Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier application or patent but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
21 October 2010 (21-10-2010)	30 December 2010 (30-12-2010)
Name and mailing address of the ISA/CA Canadian Intellectual Property Office Place du Portage I, C114 - 1st Floor, Box PCT 50 Victoria Street Gatineau, Quebec K1A 0C9 Facsimile No.: 001-819-953-2476	Authorized officer Lawrence J. Engel (819) 997-2936

INTERNATIONAL SEARCH REPORT
Information on patent family members

International application No.
PCT/CA2010/000486

Patent Document Cited in Search Report	Publication Date	Patent Family Member(s)	Publication Date
US2009252327A1	08 October 2009 (08-10-2009)	None	
US2006140401A1	29 June 2006 (29-06-2006)	US7809135B2	05 October 2010
US2005002532A1	06 January 2005 (06-01-2005)	AU2003203207A1	02 September 2003
		CA2369304A1	30 July 2003)
		US7634091B2	15 December 2009
		WO03065639A2	07 August 2003
		WO03065639A3	16 October 2003
US6668325B1	23 December 2003 (23-12-2003)	AU7957998A	25 January 1999
		CA2293650A1	14 January 1999
		CN1260055A	12 July 2000
		EP0988591A1	29 March 2000
		JP2002514333T	14 May 2002
		WO9901815A1	14 January 1999